U3-0155-SM

JC997 U.S. PTO
09/935686
08/24/01

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２０００年　８月２８日

出願番号
Application Number: 特願２０００－２５７１８４

出願人
Applicant(s): 株式会社デンソー

２００１年　７月１９日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３０６４４４２

【書類名】　特許願
【整理番号】　N000645
【提出日】　平成12年 8月28日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 9/00
【発明者】
【住所又は居所】　愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
【氏名】　神谷　政裕
【発明者】
【住所又は居所】　愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
【氏名】　江川　邦隆
【発明者】
【住所又は居所】　愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
【氏名】　松田　啓資
【発明者】
【住所又は居所】　愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
【氏名】　石原　秀昭
【特許出願人】
【識別番号】　000004260
【氏名又は名称】　株式会社デンソー
【代理人】
【識別番号】　100071135
【住所又は居所】　名古屋市中区栄四丁目６番１５号　名古屋あおば生命ビル
【弁理士】
【氏名又は名称】　佐藤　強
【電話番号】　052-251-2707
【手数料の表示】
【予納台帳番号】　008925

【納付金額】　　21,000円

【提出物件の目録】

【物件名】　　明細書　1

【物件名】　　図面　1

【物件名】　　要約書　1

【包括委任状番号】　9200169

【プルーフの要否】　　要

【書類名】　明細書

【発明の名称】　コンパイラ，記録媒体，プログラム変換装置，プログラム変換方法及びマイクロコンピュータ

【特許請求の範囲】

【請求項１】　プログラムとして記述されたソースコードからメモリ若しくはレジスタ上に配置されるビット変数の値を判定する命令と、前記ビット変数に対して即値を代入する命令とを有するＲＩＳＣ型ＣＰＵのオブジェクトコードを生成するコンパイラにおいて、

前記ソースコード中にビット演算式が記述されている場合は、そのビット演算式を演算対象である各ビット変数の値に基づく条件を判定する式に変換し、その判定式の結果が真である場合と偽である場合との夫々に対応して、前記ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードを生成することを特徴とするコンパイラ。

【請求項２】　複数のビット変数の集合をレジスタ変数として、前記ＲＩＳＣ型ＣＰＵに内蔵されるレジスタに割り当てることが可能であることを特徴とする請求項１記載のコンパイラ。

【請求項３】　請求項１または２記載のコンパイラプログラムが記憶されていることを特徴とする記録媒体。

【請求項４】　プログラムとして記述されたソースコードファイルの内容を読み出し、メモリ若しくはレジスタ上に配置されるビット変数の値を判定する命令と、前記ビット変数に対して即値を代入する命令とを有するＲＩＳＣ型ＣＰＵのオブジェクトコードファイルを生成するプログラム変換装置において、

前記ソースコード中にビット演算式が記述されている場合は、そのビット演算式を、演算対象である各ビット変数の値に基づいて条件を判定する式に変換し、その判定式の結果が真である場合と偽である場合との夫々に対応して、前記ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードを生成するコンパイラプログラムが記憶された記憶手段を備えたことを特徴とするプログラム変換装置。

【請求項５】　前記コンパイラは、複数のビット変数の集合をレジスタ変数

として、前記ＲＩＳＣ型ＣＰＵに内蔵されるレジスタに割り当てることが可能であることを特徴とする請求項４記載のプログラム変換装置。

【請求項６】　プログラムとして記述されたソースコードファイルの内容を読み出し、メモリ若しくはレジスタ上に配置されるビット変数の値を判定する命令と、前記ビット変数に対して即値を代入する命令とを有するＲＩＳＣ型ＣＰＵのオブジェクトコードファイルを生成するプログラム変換方法において、

前記ソースコード中にビット演算式が記述されている場合に、そのビット演算式を、演算対象である各ビット変数の値に基づいて条件を判定する式に変換し、その判定式の結果が真である場合と偽である場合との夫々に対応して、前記ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードを生成することを特徴とするプログラム変換方法。

【請求項７】　複数のビット変数の集合をレジスタ変数として、前記ＲＩＳＣ型ＣＰＵに内蔵されるレジスタに割り当てることを特徴とする請求項６記載のプログラム変換方法。

【請求項８】　メモリ若しくはレジスタ上に配置されるビット変数の値を判定する命令と、前記ビット変数に対して即値を代入する命令とを有するＲＩＳＣ型ＣＰＵと、

プログラムとして記述されたソースコード中に記述されているビット演算式について、その演算対象である各ビット変数の値を夫々判定し、各判定の結果に基づいて前記ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードを生成するコンパイラによってコード変換されたプログラムが記憶されているプログラムメモリとを備えたことを特徴とするマイクロコンピュータ。

【請求項９】　前記プログラムメモリに記憶されているプログラムは、ソースコード中に記述されている複数のビット変数の集合をレジスタ変数として、前記ＲＩＳＣ型ＣＰＵに内蔵されるレジスタに割り当てられるように、前記コンパイラによってオブジェクトコードが生成されていることを特徴とする請求項８記載のマイクロコンピュータ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、プログラムとして記述されたソースコードからＲＩＳＣ型ＣＰＵのオブジェクトコードを生成するコンパイラ，そのコンパイラプログラムが記憶された記録媒体，プログラム変換装置，プログラム変換方法及びマイクロコンピュータに関する。

【０００２】

【従来の技術】

現在、ＣＰＵやマイクロコンピュータなどのプログラム開発は、例えばＣ言語やＣ＋＋言語などの高級言語を用いて行われる場合が一般的である。高級言語によって記述されたソースコードは、コンパイラによってプログラム変換（コンパイル）され、ＣＰＵ等が直接実行可能な命令形式（機械語）のオブジェクトコードが生成されるようになっている。コンパイラによって生成されたオブジェクトコードは、プログラムメモリに格納され、ＣＰＵ等によって読み出されることで所期の処理が実行されるようになっている。

【０００３】

ところで、ＲＩＳＣ(Reduced Instruction Set Computer)型ＣＰＵは、機能が単純で実行時間が短い命令だけを使用可能とし、複数の命令の実行に伴うプロセスにおける各ステージ（フェッチ，デコード，実行，書込みなど）を複数の命令パイプラインによって並行に実行することでプログラム実行速度の高速化を図ったものである。そのため、ソースコードからコンパイラによって生成されるオブジェクトコードの容量は、ＣＩＳＣ(Complex Instruction Set Computer)型ＣＰＵに比較して多くなる傾向を示す。

【０００４】

図７は、従来のコンパイラによって生成されるオブジェクトコード（ニーモニックで表現）の例を示すプログラムリストである。ソースコードは、

ｂｉｔ１＝ｂｉｔ２＆ｂｉｔ３　　…（１）

であるとする。即ち、ビット変数ｂｉｔ２と、ビット変数ｂｉｔ３との論理積をとった結果をビット変数ｂｉｔ１に代入する、というビット演算である。また、

各ビット変数ｂｉｔ１～ｂｉｔ３は、８ビット変数ＢＩＴ１～ＢＩＴ３の第４ビット（ＬＳＢから第３ビット目）に配置されている変数であるとする。

また、図８は、（１）式のビット演算のソースコードをコンパイラが変換する場合のプロセスを示すフローチャートである。

【０００５】

図７において、オブジェクトコードの①部分では、例えばメモリ上に配置された８ビット変数ＢＩＴ２の内容を汎用レジスタｒｅｇ１に読み出してからｒｅｇ１を３ビット右シフトして、その第４ビットをＬＳＢに位置させる。そして、レジスタｒｅｇ１とマスクデータ＄０００１（“＄”は１６進数を示すシンボル）との論理積をとることで、レジスタｒｅｇ１のＬＳＢ以外のビットをクリアする。ここまでが、図８のステップＡ１に対応する。オブジェクトコードの②部分では８ビット変数ＢＩＴ３について同様の処理を行っており、図８のステップＡ２に対応する。

【０００６】

オブジェクトコード③部分では、レジスタｒｅｇ１，ｒｅｇ２の論理積演算を行う（図８のステップＡ３に対応）。続くオブジェクトコード④部分では、前記論理積演算の結果が格納されているレジスタｒｅｇ１を３ビット左シフトさせると、８ビット変数ＢＩＴ１の内容をレジスタｒｅｇ２に読み出し、レジスタｒｅｇ２とマスクデータ＄ｆｆｆ７との論理積演算を行うことで、代入先となるレジスタｒｅｇ２の第４ビットをクリアし、それ以外のビットは保存する。そして、レジスタｒｅｇ１，ｒｅｇ２の論理和演算を行い、その論理積演算結果が格納されているレジスタｒｅｇ２の内容をメモリ上の８ビット変数ＢＩＴ１に転送すると（図８のステップＡ４に対応）、一連の演算処理が終了する。

【０００７】

【発明が解決しようとする課題】

このように、ＲＩＳＣ型ＣＰＵでは命令数が少なくビット変数の値を読み出すための専用命令がないので、他の複数の命令の組み合わせによって所期の処理を実行するようにコード変換されている。その結果、ソースコードにおいて１ステップで記述されたビット演算を実行するのに、オブジェクトコードでは１８ステ

ップを要している。従って、ＣＰＵのプログラム記憶領域をより多く必要とすることになり、また、ステップ数が多くなった分だけ処理速度が低下するという問題があった。

【０００８】

本発明は上記事情に鑑みてなされたものであり、その目的は、プログラム容量をより小さくでき、且つ、処理速度を高速化できるようにオブジェクトコードを生成するコンパイラ、そのコンパイラプログラムが記憶された記録媒体，プログラム変換装置，プログラム変換方法及びマイクロコンピュータを提供することにある。

【０００９】

【課題を解決するための手段】

請求項１記載のＲＩＳＣ型ＣＰＵのコンパイラによれば、プログラムとして記述されたソースコード中にビット演算式が記述されている場合は、そのビット演算式を、演算対象である各ビット変数の値に基づいて条件を判定する式に変換する。そして、その判定式の結果が真である場合と偽である場合との夫々に対応して、ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードが生成される。

【００１０】

例えば、ビット演算が、ビット変数ａとビット変数ｂとの論理積演算を行った結果をビット変数ｃに代入するというビット演算、即ち、

ｃ＝ａ＆ｂ　　…（２）

であるとすると、（２）式右辺に記述されたビット演算式“ａ＆ｂ”の結果が“１”となるのは“ａ＝１”且つ“ｂ＝１”の場合であり、例えばＣ言語においてｉｆ文やｗｈｉｌｅ文などに使用される条件判定式で表現すれば、

ａ＝＝１　＆＆　ｂ＝＝１　　…（３）

となる。つまり、（２）式は、条件判定式（３）が真である場合に（２）式左辺のビット変数ｃに“１”を代入し、条件判定式（３）が偽である場合にビット変数ｃに“０”を代入することと等価であり、ビット演算式は条件判定式に置き換えることが可能である。

【００１１】

従って、ソースコード中に記述されているビット演算式については、置き換えた判定式に基づいて、各ビット変数の値を判定し、各判定の結果に基づいて条件分岐を行うなどして、前記ビット演算の結果が格納されるビット変数に異なる即値“０，１”を代入するようにオブジェクトコードを生成すれば、ビット演算と等価な処理を行うことができる。

【００１２】

即ち、ビット演算について斯様にオブジェクトコードを生成することで、ＲＩＳＣ型ＣＰＵが実行する命令のステップ数が従来よりも短くなる。そして、一般に、ＣＰＵのプログラム中においてビット演算を行う頻度は比較的高いことから、プログラム全体として生成されるオブジェクトコードの量を少なくすることができる。そして、プログラムの容量（サイズ）を従来よりも小さくすることができると共に、ＲＩＳＣ型ＣＰＵがビット演算を実行する場合の処理速度を高速化することが可能となる。

【００１３】

請求項２記載のコンパイラによれば、所謂ビットフィールドと称する複数のビット変数の集合をレジスタ変数としてＲＩＳＣ型ＣＰＵに内蔵されるレジスタに割り当てることができるので、ユーザはプログラミングを多様に行うことができ、ＲＩＳＣ型ＣＰＵによる処理を効率的に行わせることが可能となる。

【００１４】

請求項４記載のＲＩＳＣ型ＣＰＵのプログラム変換装置によれば、記憶手段に記憶されているコンパイラは、プログラムとして記述されたソースコード中にビット演算式が記述されている場合は、そのビット演算式を、演算対象である各ビット変数の値に基づいて条件を判定式に変換する。そして、その判定式の結果が真である場合と偽である場合との夫々に対応して、ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードが生成される。

【００１５】

即ち、論理積や論理和などのビット演算は、上述のように、各ビット変数の値

に基づいて条件判定を行う式に置き換えることが可能である。従って、ソースコード中に記述されているビット演算式については、置き換えた条件判定式に基づいて、各ビット変数の値を判定し、各判定の結果に基づいて条件分岐を行うなどして、前記ビット演算の結果が格納されるビット変数に異なる即値“０，１”を代入するようにオブジェクトコードを生成すれば、ビット演算と等価な処理を行うことができる。

【００１６】

そして、ビット演算について斯様にオブジェクトコードを生成することで、ＲＩＳＣ型ＣＰＵが実行する命令のステップ数が従来よりも短くなる。一般に、ＣＰＵのプログラム中においてビット演算を行う頻度は比較的高いことから、プログラム全体として生成されるオブジェクトコードの量が少なくなり、変換されたプログラムの容量を従来よりも小さくすることができる。また、ＲＩＳＣ型ＣＰＵがビット演算を実行する場合の処理速度が高速化される。

【００１７】

請求項５記載のプログラム変換装置によれば、所謂ビットフィールドと称する複数のビット変数の集合をレジスタ変数としてＲＩＳＣ型ＣＰＵに内蔵されるレジスタに割り当てるので、ユーザはプログラミングを多様に行うことができ、ＲＩＳＣ型ＣＰＵによる処理を効率的に行わせることが可能となる。

【００１８】

請求項８記載のマイクロコンピュータによれば、ＲＩＳＣ型ＣＰＵが実行するプログラムが記憶されているプログラムメモリには、ソースコード中に記述されているビット演算式については、そのビット演算式を、演算対象である各ビット変数の値に基づいて条件を判定式に変換し、その判定式の結果が真である場合と偽である場合との夫々に対応して、ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードを生成するコンパイラによってコード変換されたプログラムが記憶されている。

【００１９】

即ち、論理積や論理和などのビット演算は、上述のように、各ビット変数の値に基づいて条件判定を行う式に置き換えることが可能である。従って、ソースコ

ード中に記述されているビット演算式については、置き換えた条件判定式に基づいて、各ビット変数の値を判定し、各判定の結果に基づいて条件分岐を行うなどして、前記ビット演算の結果が格納されるビット変数に異なる即値“０，１”を代入するようにオブジェクトコードを生成すれば、ビット演算と等価な処理を行うことができる。

【００２０】

そして、ビット演算について斯様にオブジェクトコードを生成することで、ＲＩＳＣ型ＣＰＵが実行する命令のステップ数が従来よりも短くなるので、プログラム全体として生成されるオブジェクトコードの量が少なくなり、プログラムメモリの容量を従来よりも小さくすることができる。また、ＲＩＳＣ型ＣＰＵがビット演算を実行する場合の処理速度を高速化することが可能となる。

【００２１】

請求項９記載のマイクロコンピュータによれば、プログラムメモリに記憶されているプログラムは、ソースコード中に記述されている複数のビット変数（所謂ビットフィールド）をレジスタ変数として、ＲＩＳＣ型ＣＰＵに内蔵されるレジスタに割り当てられるようにコンパイラによってオブジェクトコードが生成されるので、ＲＩＳＣ型ＣＰＵによる処理を効率的に行わせることが可能となる。

【００２２】

【発明の実施の形態】

以下、本発明の一実施例について図１乃至図６を参照して説明する。図４は、プログラム変換装置の構成を示す図である。プログラム変換装置たるパーソナルコンピュータ（パソコン）若しくはワークステーション１には、コンパイラ２がインストールされている。具体的には、パソコン１の本体１ａに内蔵されているハードディスク等の記憶装置（記憶手段）にコンパイラ２のプログラムファイルが記憶されている。

【００２３】

本体１ａ内蔵の記憶装置には、図５に示すように、ユーザが例えばＣ言語などの高級言語によって記述したソースコードファイル３も記憶されている。そして、ユーザは、コンパイラ２のプログラムをパソコン１上で起動することで、ソー

スコードファイル３からオブジェクトコードファイル４を変換生成させる。コンパイラ２は、ソースコードファイル３に記述されているＣ言語のソースコードを読み出してその内容を解釈すると、ＣＰＵ等が固有の命令によって対応する処理を最も効率的に実行できるようなオブジェクトコードを生成するようにコンパイルを行う。

【００２４】

パソコン１の本体１ａにはＲＯＭライタ５が接続されており、両者は、例えばＲＳ－２３２Ｃ等のシリアル通信プロトコルによって通信が可能となっている。コンパイラ２よって生成されたオブジェクトコードファイル４はＲＯＭライタ５に転送されると、ＲＯＭライタ５にセットされたマイクロコンピュータ（マイコン）６に内蔵されているプログラムメモリ７（図６参照）にバイナリデータとして書き込まれるようになっている。

【００２５】

図６は、マイコン６の電気的構成を示す機能ブロック図である。マイコン６は、ＲＩＳＣ型のＣＰＵ８，ＥＥＰＲＯＭやフラッシュＲＯＭなどからなるプログラムメモリ７及びＳＲＡＭなどからなる内部メモリ９を備え、ワンチップマイコンとして構成されている。また、マイコン６の外部には、ＤＲＡＭなどからなる外部メモリ１０が接続されている。ＣＰＵ８と、プログラムメモリ７，内部メモリ９及び外部メモリ１０は、アドレスバス１１及びデータバス１２を介して接続されている。

【００２６】

ＣＰＵ８の内部には、演算等を行うための演算部（ＡＬＵ(Alithmetic Logical Unit)）１３，演算部１３が演算等を行う場合に使用される複数のレジスタが配置されているレジスタ部１４，及びレジスタ部１４等にロード／ストア等の制御を行うための制御部１５などが配置されている。

【００２７】

また、ＣＰＵ８は、前提構成として固有の命令に、少なくとも、

ビットテスト命令　　：　ｔｓｔ（．ｂ／ｗ／ｌ）

ビットセット命令　　：　ｓｅｔ（．ｂ／ｗ／ｌ）

ビットクリア命令　　：　ｃｌｒ（．ｂ／ｗ／ｌ）

を備えている。

【００２８】

ビットテスト命令は、指定されたビット変数の値をテスト（判定）する命令であり、そのテスト結果“０，１”は、ＣＰＵ８のレジスタ部１４内にあるコンディションコードレジスタ（ＣＣＲ）のＺフラグに制御部１５によって反映されるようになっている。即ち、テスト結果が“０”であれば、ＣＣＲのＺフラグがセットされ、テスト結果が“１”であればＺフラグはクリアされる。

【００２９】

また、このビットテスト命令は、ユーザの定義などによりビット変数がメモリに割り当てられている場合と、レジスタ部１４内の汎用レジスタに割り当てられている場合との何れであっても実行することが可能である。（．ｂ／ｗ／ｌ）は、メモリ上のビット変数をテスト等する場合のアクセスサイズを指定するものであり、（バイト（８）／ワード（１６）／ロングワード（３２））に対応する。尚、レジスタ上のビット変数をテスト等する場合は、アクセスサイズの指定は不要である。

また、ビットセット命令は、指定されたビット変数に即値“１”を代入する命令であり、ビットクリア命令は、指定されたビット変数に即値“０”を代入する命令である。

【００３０】

次に、本実施例の作用について図１乃至図３をも参照して説明する。図１は、コンパイラ２がビット演算に関するコンパイル処理を行う場合のプロセスを示すフローチャートである。先ず、コンパイラ２は、Ｃ言語で記述された右辺のビット演算式を解釈（デコード）すると、ブール代数規則に基づき結果が１（真）または０（偽）となる条件判定式に変換する。そして、その条件判定式に対応するオブジェクトコードを出力する（ステップＳ１）。

【００３１】

次に、コンパイラ２は、条件判定式が真であった場合に、ビット演算式の結果が代入される左辺のビット変数に“１”若しくは“０”（異なる即値）を代入す

るためのコードを生成し、それから、処理終了へ分岐するためのコードを生成する（ステップＳ２）。それから、条件判定式が偽であった場合に、ビット演算式の結果が代入される左辺のビット変数に“０”若しくは“１”を代入するためのコードを生成すると（ステップＳ３）と、処理終了となる。

【００３２】

ここで、図２は、図１のフローチャートに基づいて、具体的に（１）式のビット演算をコンパイル処理してオブジェクトコードを生成する場合のフローチャートであり、図３は、図２のフローチャートに従うコンパイル処理によって生成されるオブジェクトコードをニーモニックで示すプログラムリストである。

【００３３】

（１）式右辺に記述されたビット演算式“ｂｉｔ２＆ｂｉｔ３”の結果が“１”となるのは、“ｂｉｔ２＝１”且つ“ｂｉｔ３＝１”の場合であり、Ｃ言語における条件判定式で表現すると、

ｂｉｔ２＝＝１　＆＆　ｂｉｔ３＝＝１　　…（４）

となる。従って、（１）式は、条件判定式（４）が真である場合に（１）式左辺のビット変数ｂｉｔ１に“１”を代入し、条件判定式（４）が偽である場合にビット変数ｂｉｔ１に“０”を代入することと等価である。

【００３４】

そこで、図２においては、先ずビット変数ｂｉｔ２の値を判定し（ステップＢ１）、その値が“０”であればビット変数ｂｉｔ１に“０”を書き込んで（ステップＢ３）処理を終了する。また、ビット変数ｂｉｔ２の値が“１”である場合はビット変数ｂｉｔ１の値を判定し（ステップＢ２）、その値が“０”であればステップＢ３に移行し、ビット変数ｂｉｔ１の値が“１”であればビット変数ｂｉｔ１に“１”を書き込んで（ステップＢ４）処理を終了する。

【００３５】

即ち、図２のフローチャートでは、ステップＢ１，Ｂ２が図１のステップＳ１に基づく処理であり、ステップＢ１，Ｂ２の何れかよりステップＢ３に移行するケースが条件判定式（４）が偽である場合に対応する。そして、ステップＢ３は、図１のステップＳ３に基づく処理である。また、ステップＢ２よりステップＢ

４に移行するケースが条件判定式（４）が真である場合に対応し、ステップＢ４は、図１のステップＳ２に基づく処理である。

【００３６】

次に、実際のコンパイル処理結果のプログラムリストである図３を参照すると、ステップ①でビット変数ｂｉｔ２の値を判定し、ステップ②ではその値が“０”であればラベルＬＯＯ１に分岐させる（ステップＢ１に対応）。例えば、ビット変数ｂｉｔ２の値が“０”であれば、前述したように、ＣＰＵ８のレジスタ部１４内にあるＣＣＲのＺフラグがセットされ、ステップ②での条件判定は真となる。

【００３７】

次のステップ③，④では、ビット変数ｂｉｔ３に関してステップ①，②と同様の処理を行う（ステップＢ２に対応）。以上のステップ①～④が図１のステップＳ１を実行した結果であり、条件判定式（４）に対応して出力されたオブジェクトコードとなる。そして、ステップ②，④の何れかにおいてラベルＬＯＯ１に分岐するケースが、条件判定式の結果が偽の場合に対応する。また、ステップ②，④の何れにおいてもラベルＬＯＯ１に分岐することなく、次のステップ⑤を実行するケースが、条件判定式の結果が真の場合に対応する。

【００３８】

条件判定式の結果が真の場合に対応するステップ⑤では、ビット変数ｂｉｔ１に“１”をセットし、このビット演算処理を終了するためにラベルＬＯＯ２に分岐する（ステップ⑥）。この部分が図１のステップＳ２を実行した結果であり、図２ではステップＢ４に対応する。

また、条件判定式の結果が偽の場合に対応するステップ⑦では、ビット１をゼロクリアして処理を終了する。この部分が図１のステップＳ３を実行した結果であり、図２ではステップＢ３に対応する。

【００３９】

斯様にコンパイラ２によって生成されたオブジェクトコードのサイズは、図７に示す従来のコンパイラによって生成されたコードのサイズに比較すると大幅に縮小されている。

【００４０】

以上のようにコンパイラ２によってコンパイル処理が行われた結果生成されたオブジェクトコードファイル４は、上述のようにパソコン１よりＲＯＭライタ５に転送され、ＲＯＭライタ５にセットされたマイコン６内蔵のプログラムメモリ7にバイナリデータとして書き込まれる。

【００４１】

そして、マイコン６のＣＰＵ８は、プログラムメモリ７に記憶されたオブジェクトコードをプログラムとして読み出して実行する。尚、ビット変数ｂｉｔ１～ｂｉｔ３は、ユーザの定義により内部メモリ９，外部メモリ１０またはレジスタ部１４内の汎用レジスタなどの何れに配置されているものでも構わない。ビット変数が汎用レジスタ上に割り付けられている場合は、例えば、図３のステップ①における“ｔｓｔ．ｂ　ｂｉｔ２”に代えて、“ｔｓｔ　ｂｉｔ２”がオブジェクトコードとして生成される。

【００４２】

ところで、例えば、ビット変数ｂｉｔ２が内部メモリ９上に配置されている場合、図３のステップ①における“ｔｓｔ．ｂ　ｂｉｔ２”を実行すると、ＣＰＵ８の演算部１３は、内部メモリ９よりビット変数ｂｉｔ２の内容を読み出して判定を行う。この場合、例えば、図６に示すコードの“ｍｏｖ　ＢＩＴ２，ｒｅｇ１”などの命令が１クロック（ＣＰＵ８の動作クロック）で実行可能であるとすると、“ｔｓｔ．ｂ　ｂｉｔ２”の実行には２～３クロックを要することになるが、コードサイズの全体が大幅に縮小されているので、トータルでの処理時間は短縮される。

【００４３】

以上のように本実施例によれば、パソコン１の本体１ａにインストールされたコンパイラ２は、プログラムとして記述されたソースコードファイル３のソースコード中に記述されているビット演算式を、演算対象である各ビット変数ｂｉｔ２，ｂｉｔ３の値に基づいて条件を判定する式に変換し、その判定式の結果が真である場合と偽である場合との夫々に対応して、ビット演算の結果が格納されるビット変数ｂｉｔ１に異なる即値“０，１”を夫々代入する処理を行うようにオ

ブジェクトコードを生成し、ＲＩＳＣ型ＣＰＵ８に最適なオブジェクトコードファイル４を出力するようにした。

【００４４】

従って、ＣＰＵ８が実行する命令のステップ数が従来よりも短くなり、プログラム全体として生成されるオブジェクトコードの量が少なくなるので、プログラムの容量を従来よりも小さくすることができると共に、ＣＰＵ８がビット演算を実行する場合の処理速度を高速化することが可能となる。

【００４５】

また、マイコン６のプログラムメモリ７には、コンパイラ２によってコード変換されたプログラムが記憶され、マイコン６のＣＰＵ８は、プログラムメモリ７に記憶されたオブジェクトコードをプログラムとして読み出して実行するので、ＣＰＵ８が実行する命令のステップ数が従来よりも短くなって、プログラム全体として生成されるオブジェクトコードの量が少なくなり、プログラムメモリ７の容量を従来よりも小さくすることができる。そして、ＣＰＵ８がビット演算を実行する場合の処理速度が高速化されるので、マイコン６の処理能力を向上させることができる。

【００４６】

本発明は上記し且つ図面に記載した実施例にのみ限定されるものではなく、次のような変形または拡張が可能である。

ビット演算式は、論理積に限ることなく、論理和や排他的論理和などを行うものでも良い。その場合、論理の違いに応じて、条件分岐コマンド“ｂｅｑ”に代えて“ｂｎｅ（ｎｏｔ　ｅｑｕａｌ）”などをオブジェクトコードとして生成すれば良い。

また、ソースコードにおいて複数のビット変数の集合，所謂ビットフィールドがレジスタ変数としてＣＰＵ８のレジスタ部１４に割り当てている場合であっても、コンパイラ２は、ビットテスト命令によってビットフィールドの各ビット変数毎に値の判定を行うようにコードを出力する。従って、ユーザはプログラミングを多様に行うことができ、ＣＰＵ８による処理を効率的に行わせることが可能となる。

【００４７】

プログラムメモリ７に対するプログラムデータの書き込みは、ＲＯＭライタ５を使用するものに限らない。例えば、マイコン６側にＲＳ－２３２Ｃ等のシリアル通信インターフェイスを設けて、パソコン１側よりマイコン６にプログラムデータを直接送信することで、ＣＰＵ８によってプログラムデータをプログラムメモリ７に書き込ませるようにしても良い。

コンパイラ２のプログラムを、ＣＤ－ＲＯＭ，フロッピーディスクやＭＯディスク等の記録媒体に記憶させても良く、また、そのような記録媒体に記憶されている状態で、パソコン１の主記憶上にプログラムを読み出してコンパイル処理を行っても良い。

マイクロコンピュータとしては、マイコン６のようなワンチップマイコンに限ることなく、ＣＰＵ８とプログラムメモリ７とが夫々独立のチップとして形成されて同一のプリント基板上に搭載されているようなマイクロコンピュータであっても良い。この場合、ＲＯＭライタ５には、プログラムメモリ７のみを接続してプログラムデータを書き込むようにすれば良い。

また、プログラムメモリは、書き替え可能なメモリに限らず、例えばマスクＲＯＭなどの書き替え不可能なメモリを用いても良い。

【図面の簡単な説明】

【図１】

本発明の一実施例であり、コンパイラがビット演算に関するコンパイル処理を行う場合のプロセスを示すフローチャート

【図２】

図１のフローチャートに基づいて、コンパイラが（１）式のビット演算をコンパイル処理してオブジェクトコードを生成する場合のフローチャート

【図３】

図２のフローチャートに従うコンパイル処理によって生成されるオブジェクトコードをニーモニックで示すプログラムリスト

【図４】

プログラム変換装置たるパーソナルコンピュータ若しくはワークステーション

の構成を示す図

【図５】

コンパイラが、ソースコードファイルからオブジェクトコードファイルを変換生成する処理を概念的に示す図

【図６】

マイクロコンピュータの電気的構成を示す機能ブロック図

【図７】

従来技術を示す図３相当図

【図８】

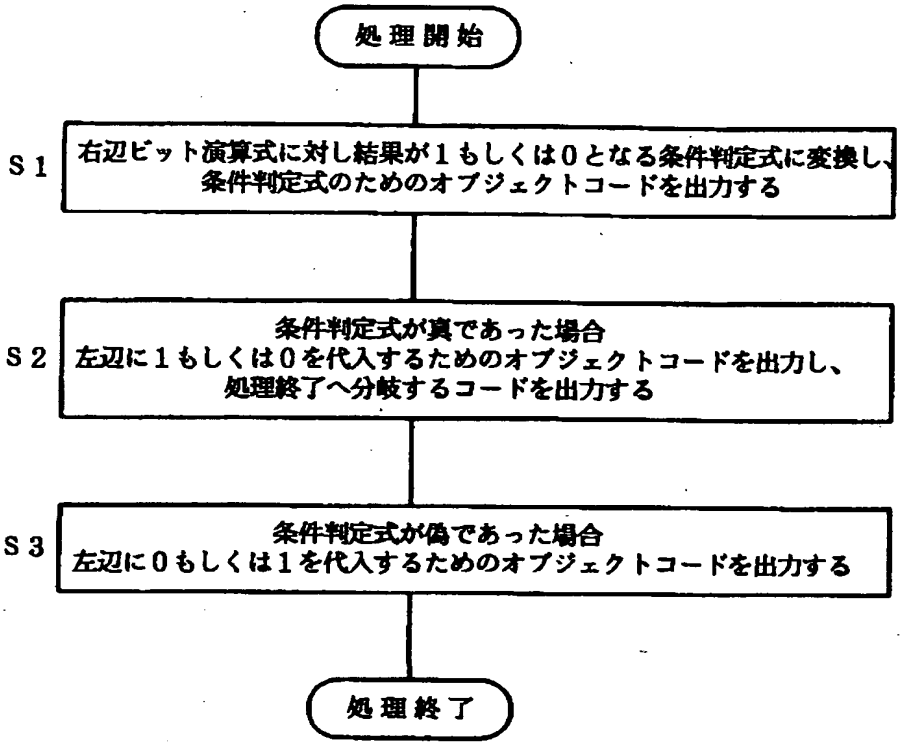
（１）式のビット演算のソースコードを、従来のコンパイラが変換する場合のプロセスを示すフローチャート

【符号の説明】

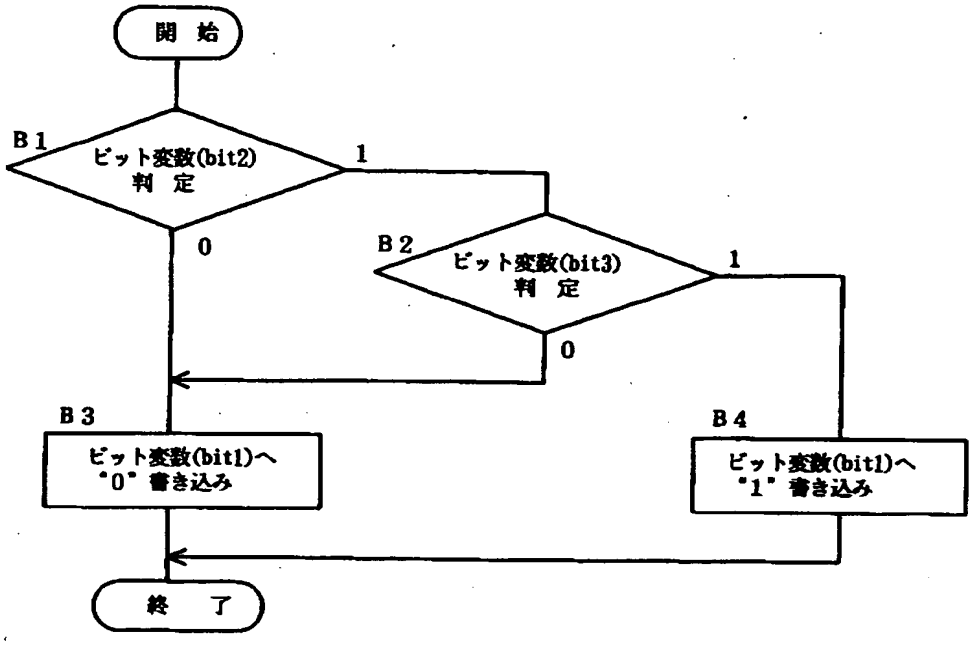
１はパーソナルコンピュータ若しくはワークステーション（プログラム変換装置）、２はコンパイラ、３はソースコードファイル、４はオブジェクトコードファイル、６はマイクロコンピュータ、７はプログラムメモリ、８はＣＰＵを示す。

【書類名】　図面

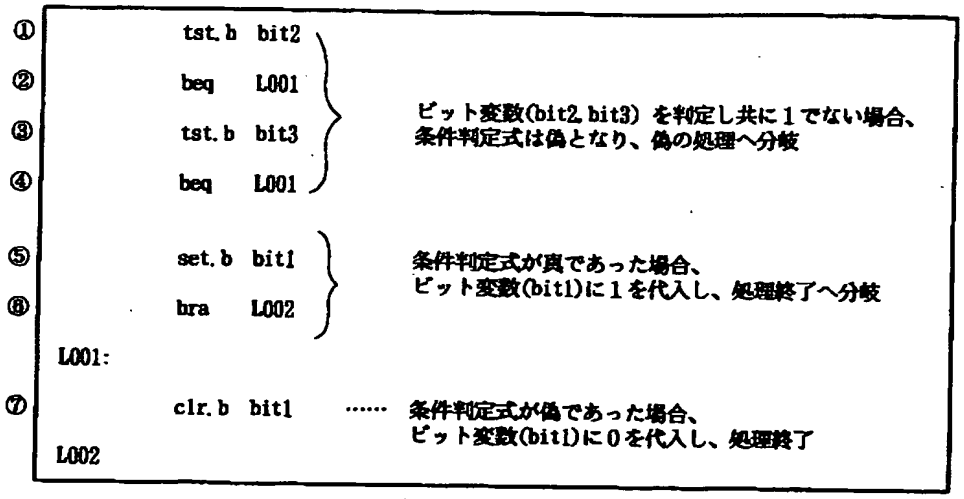
【図１】


処理開始
S１
右辺ビット演算式に対し結果が１もしくは０となる条件判定式に変換し、
条件判定式のためのオブジェクトコードを出力する
S２
条件判定式が真であった場合
左辺に１もしくは０を代入するためのオブジェクトコードを出力し、
処理終了へ分岐するコードを出力する
S３
条件判定式が偽であった場合
左辺に０もしくは１を代入するためのオブジェクトコードを出力する
処理終了

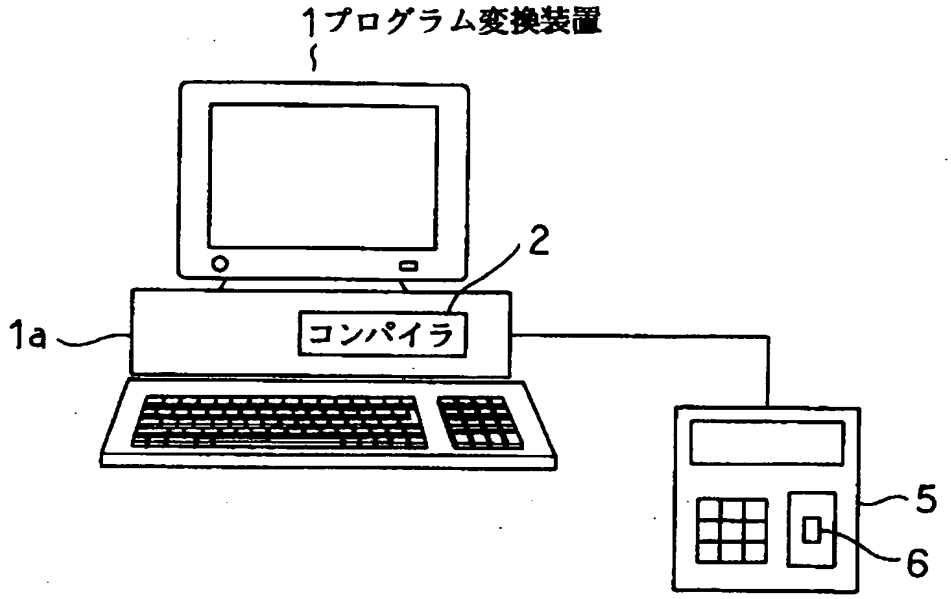
【図２】


開 始
B 1
ビット変数(bit2)
判 定
1
0
B 2
ビット変数(bit3)
判 定
1
0
B 3
ビット変数(bit1)へ
"0" 書き込み
B 4
ビット変数(bit1)へ
"1" 書き込み
終 了

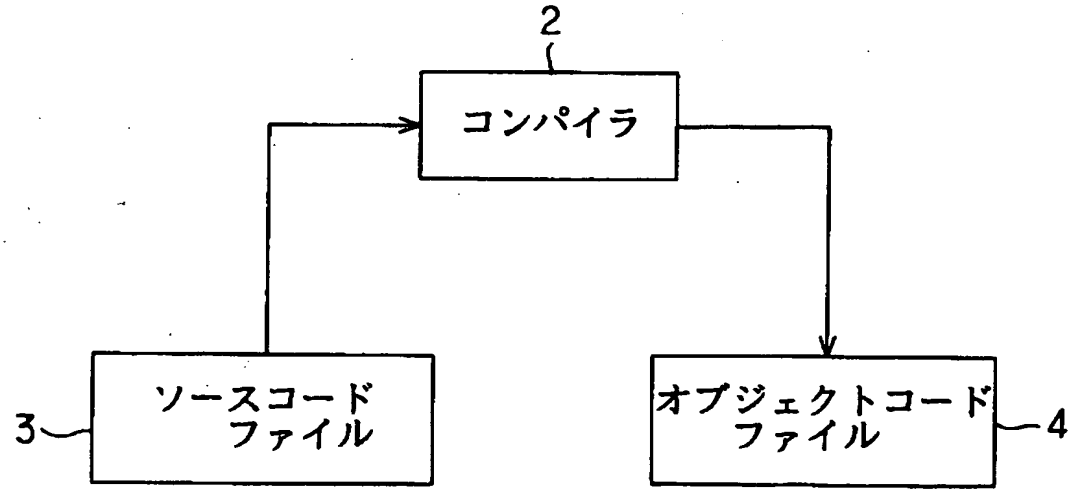
【図３】


①
tst.b bit2
②
beq L001
③
tst.b bit3
④
beq L001
ビット変数(bit2, bit3) を判定し共に１でない場合、
条件判定式は偽となり、偽の処理へ分岐
⑤
set.b bit1
⑥
bra L002
条件判定式が真であった場合、
ビット変数(bit1)に１を代入し、処理終了へ分岐
L001:
⑦
clr.b bit1 ……
条件判定式が偽であった場合、
ビット変数(bit1)に０を代入し、処理終了
L002

【図４】

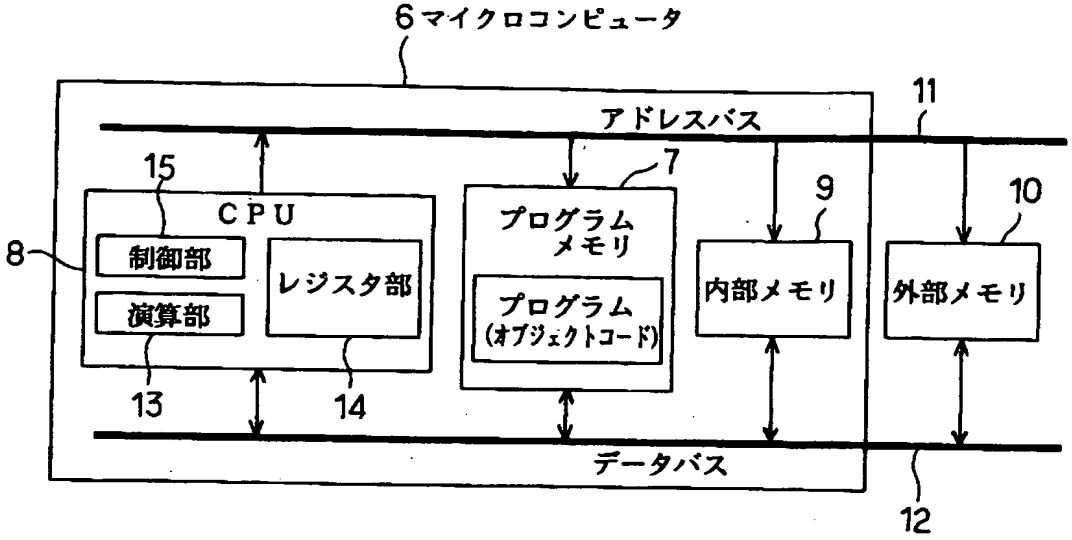


【図５】

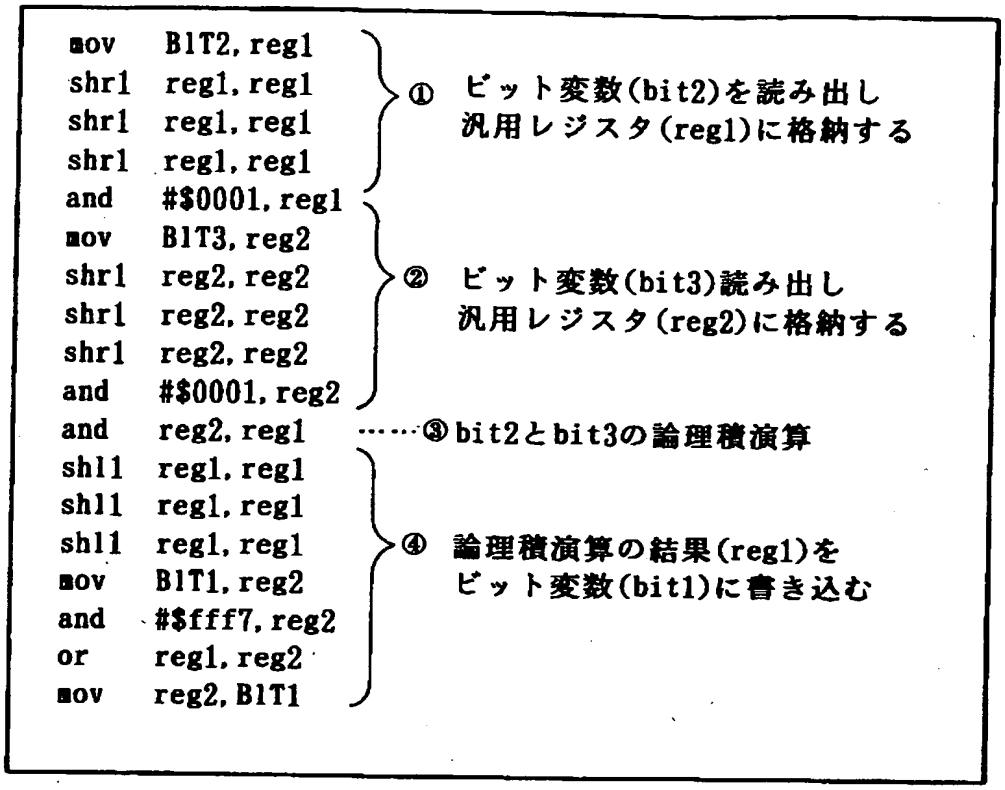


出証特２００１－３０６４４４２

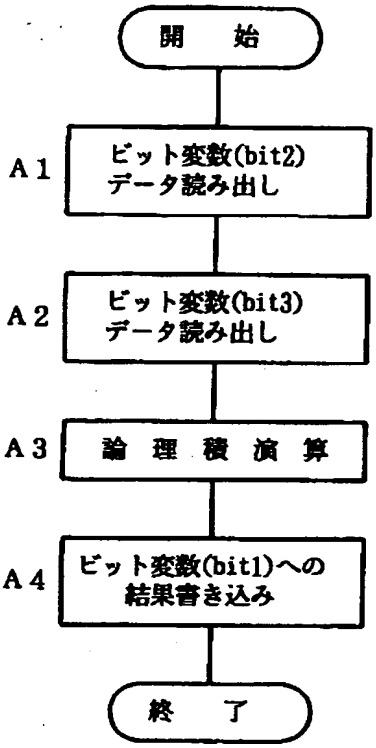
【図６】


6 マイクロコンピュータ
11
アドレスバス
15
CPU
制御部
演算部
レジスタ部
8
13
14
7
プログラム
メモリ
プログラム
(オブジェクトコード)
9
内部メモリ
10
外部メモリ
データバス
12

【図７】


mov BIT2, reg1
shr1 reg1, reg1
shr1 reg1, reg1
shr1 reg1, reg1
and #$0001, reg1
① ビット変数(bit2)を読み出し
汎用レジスタ(reg1)に格納する
mov BIT3, reg2
shr1 reg2, reg2
shr1 reg2, reg2
shr1 reg2, reg2
and #$0001, reg2
② ビット変数(bit3)読み出し
汎用レジスタ(reg2)に格納する
and reg2, reg1
……③bit2とbit3の論理積演算
sh11 reg1, reg1
sh11 reg1, reg1
sh11 reg1, reg1
mov BIT1, reg2
and #$fff7, reg2
or reg1, reg2
mov reg2, BIT1
④ 論理積演算の結果(reg1)を
ビット変数(bit1)に書き込む

【図８】


開　始
A1
ビット変数(bit2)
データ読み出し
A2
ビット変数(bit3)
データ読み出し
A3
論　理　積　演　算
A4
ビット変数(bit1)への
結果書き込み
終　了

【書類名】　要約書

【要約】

【課題】　プログラム容量をより小さくでき、且つ、処理速度を高速化できるようにオブジェクトコードを生成するコンパイラを提供する。

【解決手段】　コンパイラは、プログラムとして記述されたソースコードファイルのソースコード中に記述されているビット演算式を、演算対象である各ビット変数ｂｉｔ２，ｂｉｔ３の値に基づいて条件を判定する式に変換し（ステップＳ１）、その判定式の結果が真である場合と偽である場合との夫々に対応して、ビット演算の結果が格納されるビット変数ｂｉｔ１に異なる即値“０，１”を夫々代入する処理を行うようにオブジェクトコードを生成し（ステップＳ２，Ｓ３）、ＲＩＳＣ型ＣＰＵに最適なオブジェクトコードファイルを出力する。

【選択図】　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００４２６０〕

１．変更年月日　　１９９６年１０月　８日
［変更理由］　名称変更
住　所　　愛知県刈谷市昭和町１丁目１番地
氏　名　　株式会社デンソー